Deep Network with Support Vector Machines

Sangwook Kim, Swathi Kavuri, and Minho Lee*

School of Electronics Engineering, Kyungpook National University,
1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, South Korea
sangwook@ee.knu.ac.kr,{swati.kavuri,mholee}@gmail.com

Abstract. Deep learning methods aims at learning features automatically at multiple levels that allow the system to learn complex functions mapping the input to the output directly from data. This ability to automatically learn powerful features will become increasingly important as the amount of data and range of applications to machine learning methods continues to grow. In this context we propose a deep architecture model using Support Vector Machine (SVM) which has inherent ability to select data points important for classification with good generalization capabilities. Since SVM can effectively discriminate features, we used support vectors with kernel as non-linear discriminant features for classification. By stacking SVMs in to multiple layers, we can obtain deep features without extra feature engineering steps and get robust recognition accuracy. Experimental results show that the proposed method improves generalization performance on Wisconsin Breast Cancer dataset.

Keywords: pattern recognition, deep learning, support vector machine.

1 Introduction

Classification results of learning algorithms are inherently limited in performance by the features extracted [1]. Deep learning is required to learn complicated function that can represent higher level extractions. Deep learning architectures consist of multiple intermediate layers rather than a single hidden layer, and adequate learning algorithm to train those layers. For the deep learning, multiple layers are expected to replace manual domain-specific feature engineering [2]. Also, recent neuroscience researches have provided backgrounds to deep feature extraction [1]. Besides the early attentions to the importance of deep architecture [3,4], deep learning was not prevalent since there was no effective learning method applicable for existing learning machines except few models [5,6]. Restricted Boltzmann Machine (RBM) is a generative stochastic neural network that can learn a probability distribution over its set of inputs and initially, was invented by Smolensky in 1986 [7]. But as G. Hinton et al. proposed the RBM network with contrastive divergence [8], deep architectures using RBM network become popular for many pattern recognition and machine learning application and start to win prizes at several pattern recognition competitions without complex manual feature engineering. Although well-trained RBM networks show

* Corresponding author.

M. Lee et al. (Eds.): ICONIP 2013, Part I, LNCS 8226, pp. 458–465, 2013.

good performance, the learning algorithm requires setting user determined meta-parameters such as the learning rate, the momentum, the weight-cost, the sparsity target, the initial values of the weights, the number of hidden units and the size of each mini-batch [9]. Without careful considerations or optimal engineering of these parameters the training would not be performed well or easily fall into the over-fitting problem. This gives an unsuccessful generalization performance.

Support Vector Machine is the supervised machine learning algorithm and is proposed by Vladimir N. Vapnik [10]. SVM is widely adopted for classification and regression especially with kernel trick which makes predictions for new inputs depend only on the kernel function evaluated at a sparse subset of the training data points [11]. SVM constructs maximal-margin hyper-plane which discriminates different patterns efficiently and maximal-margin enables the high generalization performance since the generalization errors can be bounded in terms of margins.

In this paper, we propose deep learning algorithm with support vector machines which allow the training of layer by layer learning. By stacking SVM, we can extract high-order discriminative features with support vectors which maximize the margin and guarantees generalization performance. The proposed method for deep learning requires setting a few user determined parameters such as the number of layers which improves its applicability to various domains of engineering.

The rest of this paper is organized as follows. In Section 2, we describe the structure and the algorithm of the proposed model. In Section 3 we present the experimental results to evaluate the performance of the proposed. Finally, we draw our conclusions in Section 4.

2 Proposed Model

2.1 Support Vector Machine

Support vector machine is the supervised learning method and widely used in classification and regression tasks. For the linearly separable problem, SVM obtains maximal-margin hyper-plane and the distance from the hyper-plane to the nearest data points on each side is maximized.

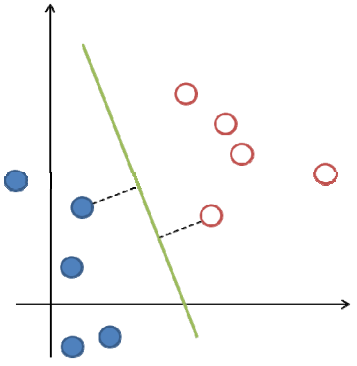

Fig. 1. Example of maximal-margin hyper-plane

Support vectors of SVM indicate the closest training data points to the hyper-plane. Binary classification problem using linear models can be represented as:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{1}$$

where $\mathbf{w}$ denotes the weight vector, $\mathbf{x}$ is input vector and b is a bias parameter of the linear decision function. SVM weights represent the importance of the corresponding input or feature for the classification.

The training data set which consists of N input patterns is represented as $\mathbf{x}_1, \ldots, \mathbf{x}_N$, and corresponding target can be expressed as $t_1, \ldots, t_N$ where $t_i \in \{-1, 1\}$, $i=1, \ldots, N$. The target label of new data point $\mathbf{x}$ is predicted with the sign of $y(\mathbf{x})$. Then if the training data set is linearly separable and the model is trained to classify training data correctly, $t_i y(\mathbf{x}_i)>0$ for all training data. The training of the parameters of SVM is constrained optimization problem which minimizes:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} a_i\{t_i(\mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b) - 1\}, \tag{2}$$

where $\mathbf{a}=(a_1,\ldots,a_N)^{\mathrm{T}}$ and a_i are Lagrange multiplier [10]. To get the optimum of above problem, we can set the derivatives with respect to $\mathbf{w}$ and b equal to zero. Then we get

$$\mathbf{w} = \sum_{i=1}^{N} a_i t_i \mathbf{x}_i, \tag{3}$$

$$0 = \sum_{i=1}^{N} a_i t_i. \tag{4}$$

And the dual representation of the above problem in Eq. (2) can be derived as the maximization problem:

$$L(\mathbf{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j t_i t_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{5}$$

with respect to a subject to the constraints

$$a_i \geq 0, i = 1, \ldots, N, \tag{6}$$

$$\sum_{i=1}^{N} a_i t_i = 0. \tag{7}$$

where a_i are obtained by training the SVM and $K(\mathbf{x}, \mathbf{x}')$ is the linear inner product of $\mathbf{x}$ and $\mathbf{x}$'. Support vectors are the training samples whose corresponding a_i values are nonzero. In this representation, weights disappear and the problem only depends on the set of a_i, t_i and $\mathbf{x}_i$. Since this is convex optimization problem, any local optimum is global optimum. Once we get a set of a_i, $\mathbf{w}$ and b are calculated using (3) and (4), respectively. With these parameters, a new data can be tested using equation (1). By substituting $\mathbf{w}$ in Eq. (1) using Eq. (3), the equation for classification of new data $\mathbf{x}$ become

$$y(\mathbf{x}) = \sum_{i=1}^{N} a_i t_i K(\mathbf{x_i}, \mathbf{x}) + b. \tag{8}$$

Because the optimization of Eq. (5) satisfies the Karush-Kuhn-Tucker (KKT) conditions, $a_i=0$ or $t_i y(\mathbf{x}_i)=1$ for every data point. Then only points which have $a_i>0$ affects

the classification and these data points are support vector. Since those points satisfies $t_i y(\mathbf{x}_i)=1$, so they are laid on maximum margin hyper-plane. SVM formulation therefore depends on the support vectors to define boundaries, which are the sample points at the discriminatory boundary and within the separating margins of the two classes.

2.2 Deep Network with Support Vector Machine

SVM obtains maximal margin hyper-plane to achieve high generalization performance. In this subsection, deep architecture with support vectors is explained. As described in the introduction section, multiple layers of deep architecture have a role to extract high-order features for recognition. Deep architecture with SVM is also used to capture features which are helpful to discriminative patterns.

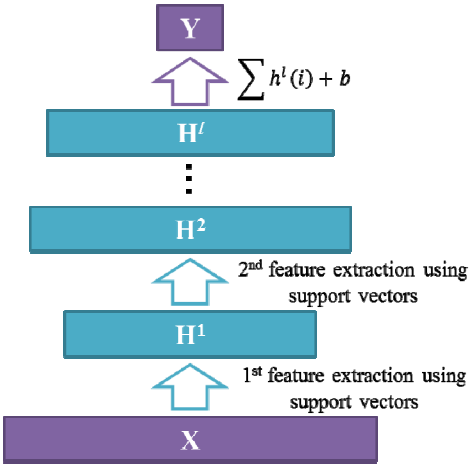

Fig. 2. The structure of the proposed model

Fig. 2 shows the structure of the proposed model. There are multiple hidden layers and each hidden layer extract discriminative features using support vectors of previous layer and corresponding multipliers. In previous subsection, linearly separable problem is solved with linear inner product of support vector and input vector. When we use the non-linear kernel instead of the inner product operation, SVM can discriminate linearly non-separable patterns also. Gaussian RBF (Radius Basis Function) kernel is most popular and represented as:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\acute{o}^2}\right). \tag{9}$$

Feature extraction is represented in detail in Fig. 3.

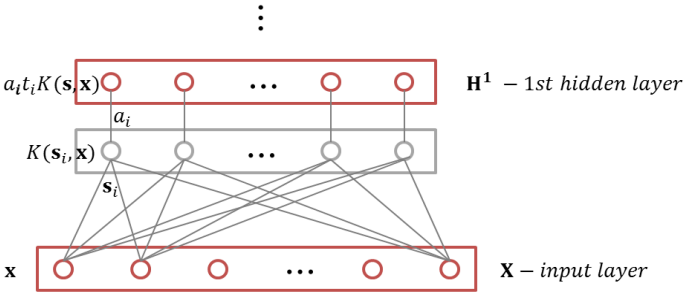

Fig. 3. Feature extraction between layers

The training of the first weight set is extracted from original training data. Let *m*-dimensional *N* input patterns of training data are $\mathbf{x}_1, \ldots, \mathbf{x}_N$. From the training data we get *m*-dimensional *p* support vectors $\mathbf{s}_1, \ldots, \mathbf{s}_p$ and corresponding Lagrange multipliers $a_1, \ldots, a_p$ and target labels $t_1, \ldots, t_p$as described in previous subsection 2.1. The activation values of the next layer are calculated as:

$$\mathbf{h}^1(i) = a_i t_i K(\mathbf{s}_i, \mathbf{x}), \tag{10}$$

where $\mathbf{h}^1(i)$ is the *i*-th element of the first hidden layer. Here *b* is not used except at the final layer since *b* is a just bias for the classification and it does not affect the intrinsic distribution of the data. The dimensionality of $\mathbf{h}^1$ is *p*, the number of support vectors of the input layer. The training of the weight connecting the input layer with the first hidden layer is completed, and weights of the next layers are trained layer by layer in the same manner. However, the training data for the next layers is the feature data transformed by previous layers rather the original data points. By feature extraction between the first layer and the next layer, original data $\mathbf{x} \in \mathbf{R}^m$ is transformed to $\mathbf{h}^1 \in \mathbf{R}^p$. For example, to train the second hidden layer, we can obtain the *p*-dimensional *N* training data $\mathbf{h}^1{}_1, \ldots, \mathbf{h}^1{}_N$ by projecting original data $\mathbf{x}_1, \ldots, \mathbf{x}_N$ on the transformed feature space whereas target values $t_1, \ldots, t_N$ are not changed. With these $\mathbf{h}^1{}_1, \ldots, \mathbf{h}^1{}_N$ and $t_1, \ldots, t_N$, the next layer also could be trained. Training procedures can be represented in algorithm 1.

Algorithm 1. Training of the Deep SVM
INPUT:
$\mathbf{X}^1$:={$\mathbf{x}_1$, …, $\mathbf{x}_N$} is training data
t :={t_1, …, t_N} is the set of corresponding target labels
K() := the RBF kernel function
n_layers := the number of layers

PROCEDURE:
FOR i=1:n_layers
{$\mathbf{S}^i$, $\mathbf{a}^i$, b^i} <= SVM($\mathbf{X}^i$,**t**)

```
  t_s^i := target labels corresponding S^i
  p := the number of support vectors S^i

  FOR j=1:N
    X^(i+1)_j <= {t_s^i_1*a^i_1*K(S^i_1,X^i_j),…, t_s_p*a^i_p*K(S^i_p,X^i_j)}
  END
END
```

OUTPUT:
S, a, t_s, b

To test the new data point, the input vector is given to the first layer and which is mapped to the next layer using Eq. (10). At the highest layer, final classification is decided by using the sign of function

$$y(\mathbf{x}) = \sum_{i=1}^{l} a_i t_i K(\mathbf{s}_i, \ddot{o}(\mathbf{x})) + b, \tag{11}$$

where $\mathbf{s}_i$ is i-th support vector, l is the number of support vectors of the final layer, and $\ddot{o}(\mathbf{x})$ represents transformed feature of test data $\mathbf{x}$ by hidden layers.

3 Experimental Results

The recognition performance of the proposed model is tested on Wisconsin Breast Cancer Database [12]. This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The data samples contain 9 attributes of Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli and Mitoses. Each attribute has the integer value from 1 to 10. Each sample has the class of benign or malignant. The dimensionality of data patterns is 9 and the number of data samples is 367. Among 367 data samples, 200 samples are benign and 157 are malignant. In our experiment, the training set consists of randomly selected 100 and 84 samples and the test set is remaining 100 and 83 samples for benign and malignant cancer respectively. Sigma of the RBF kernel parameter of SVM is set to 5 and the number of layers is 3. Table 1 shows the number of support vectors and classification accuracies.

Table 1. Recognition performance

	1st layer	2nd layer	3rd layer
Dimensionality	34	25	31
Training accuracy	100 %	100 %	100 %
Test accuracy	91.2 %	92.3 %	95.6 %

Table 1 shows the perfect accuracy on training data. There are some errors on the test set. But, test accuracy is increased as layers are stacked. This demonstrates the superior generalization performance of the proposed model to shallow network.

Table 2. Performance comparison with other models

	SVM	Deep SVM	Denoising auto-encoder
Training accuracy	100 %	100 %	100 %
Test accuracy	91.2 %	95.6 %	91.9 %

Table 2 shows the comparison of classification accuracies with several models. To test the accuracy of denoising auto-encoder [13], 3 hidden layers are used and each layer from bottom to top hidden layer has 34, 25 and 31 nodes respectively. For the training of denoising auto-encoder, mini-batch algorithm is used to make the learning efficient and 200 epochs are done to train the network. The result shows the superior generalization performance of the proposed model.

4 Conclusion

In this paper, we proposed the deep network to achieve discriminative power of high-order feature space by stacking SVMs as layers. Hidden layers use support vector and its corresponding multiplier to extract features of the input vector. Experimental results tested on Wisconsin Breast Cancer Database demonstrate that although the test performance on training set is perfect on every layer, higher layer's SVMs show better generalization performance, i.e., they have better classification accuracies on test data than lower layers. In future works, we would test the system on high-dimensional data and try to implement the incremental learning of the architecture. Error-based fine-tuning of weights also can be considered.

Acknowledgement. This research was supported by the Industrial Strategic Technology Development Program (10044009) (50%) and the R&D program (10041826) (50%) funded by the Korea Ministry of Knowledge Economy (MKE) and the Korea Evaluation Institute of Industrial Technology (KEIT).

References

1. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning-A new frontier in artificial intelligence research. IEEE Computational Intelligence Magazine 5, 13–18 (2010)
2. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2 (2009)
3. Utgoff, P.E., David, J.S.: Many-layered learning. Neural Computation 14, 2497–2529 (2002)

4. Tesauro, G.: Practical issues in temporal difference learning. Reinforcement Learning, pp. 33–53. Springer US (1992)
5. LeCun, Y., et al.: Backpropagation applied to handwritten zip code recognition. Neural computation 1, 541–551 (1989)
6. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36, 193–202 (1980)
7. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory (1986)
8. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural Computation 18, 1527–1554 (2006)
9. Hinton, G.E.: A practical guide to training restricted Boltzmann machines. Momentum 9 (2010)
10. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20, 273–297 (1995)
11. Bishop, C.M.: Pattern recognition and machine learning. Springer, New York (2006)
12. Bennett, K.P., Mangasarian, O.L.: Robust linear programming discrimination of two linearly inseparable sets. Optimization Methods and Software 1, 23–34 (1992)
13. Vincent, P., et al.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research 9999, 3371–3408 (2010)